Project Report: StudySync

Team Members:
Caroline Ribeiro (NMEC: 106093)
Mateus da Fonte (NMEC: 106082)
Victor Milhomem (NMEC: 128660)

June 11, 2025

Contents

1 Project Identification

This project was developed as part of the course *Introdução à Computação Móvel*. Below are the details of the team:

- **Project Name:** StudySync
- **Team Members:**
  - Caroline Ribeiro (NMEC: 106093)
  - Mateus da Fonte (NMEC: 106082)
  - Victor Milhomem (NMEC: 128660)

2 Introduction

2.1 Motivation

In today's fast-paced professional environment, efficient meeting management and accurate attendance tracking remain significant challenges. Traditional methods often suffer from inefficiencies, including manual sign-ins that are prone to error and time consumption, difficulties in verifying genuine presence, and the rigid reliance on internet connectivity for sharing critical meeting documents. These limitations can lead to reduced productivity, disputes over attendance records, and hindered collaboration, especially in scenarios where network access is unreliable or unavailable.

Our project addresses these prevailing issues by introducing an an innovative mobile application designed to streamline the entire meeting lifecycle. The primary motivation behind this endeavor is to enhance the reliability and convenience of meeting participation and record-keeping. We aim to eliminate the ambiguities associated with attendance by leveraging advanced sensor technology:

- **Accelerometer data** provides a robust method for verifying active participation during the meeting, ensuring attendees are genuinely present and engaged.
- **Precise location verification** against the scheduled meeting venue guarantees that attendance is recorded only for individuals physically at the designated location, thereby preventing fraudulent check-ins.

Furthermore, recognizing the critical need for uninterrupted information exchange, particularly in environments with limited or no internet access, our application integrates a **Bluetooth-based document sharing feature**. This

allows participants to securely and seamlessly exchange documents during a meeting, fostering real-time collaboration regardless of network availability.

Finally, to encourage consistent and punctual attendance, we incorporate a **gamification system**. By rewarding users for their presence, the application transforms a routine task into an engaging experience, thereby fostering a culture of punctuality and active participation. This holistic approach not only resolves existing pain points in meeting management but also introduces novel functionalities that promote a more efficient, verifiable, and interactive meeting experience.

3 Solution Overview

3.1 Requirements and Features

3.1.1 Functional Requirements

The application shall provide the following core functionalities:

2.1.1 User Management and Meeting Scheduling

- **FR1.1 User Registration and Login:** The system shall allow users to register for an account and log in securely.
- **FR1.2 Meeting Creation:** Users shall be able to create new meetings, specifying details such as:
  - Meeting title and description
  - Date and time
  - Location (with GPS coordinates for verification)
  - Invited participants (from a contact list or by email)
  - Optional agenda
- **FR1.3 Meeting Invitation and Notification:** The system shall send invitations and notifications to invited participants.
- **FR1.4 Meeting Viewing:** Users shall be able to view details of their scheduled meetings (upcoming and past).

2.1.2 Attendance Marking and Verification

- **FR2.1 Location-Based Check-in:** The application shall allow users to check in to a meeting only if their current GPS location is within a predefined radius of the meeting's designated location.
- **FR2.2 Accelerometer-Based Presence Verification (Planned):** This feature is under development. The application aims to use accelerometer data to detect active presence during meetings, enhancing the integrity of attendance.

- **FR2.3 Manual Override for Attendance (Admin/Organizer):** Meeting organizers shall have the ability to manually mark attendance for participants in exceptional cases (e.g., technical issues).
- **FR2.4 Attendance History:** Users shall be able to view their attendance history for past meetings.

2.1.3 Offline Document Sharing (Planned) The following features are planned for future development:

- **FR3.1 Bluetooth Document Transmission:** Participants will be able to send and receive documents (e.g., PDFs, images, text files) via Bluetooth, even without internet connectivity.
- **FR3.2 Document Reception Notification:** Users will be notified when documents are received.
- **FR3.3 Document Storage:** Documents will be stored locally for offline access.
- **FR3.4 Document Browse:** Users will be able to browse and open received files within the application.

2.1.4 Gamification System

- **FR4.1 Presence Points:** Users shall earn points for each successfully marked attendance at a meeting.
- **FR4.2 Punctuality Bonuses:** Additional points shall be awarded for punctual attendance.
- **FR4.3 Streaks:** The system shall track consecutive attendance streaks and reward users for maintaining them.
- **FR4.4 Leaderboard:** A leaderboard shall display top-ranking users based on their accumulated gamification points.
- **FR4.5 Achievements/Badges (Planned):** A badge system for attendance milestones (e.g., "Perfect Attendee") is planned but not yet implemented.

3.1.2 Non-Functional Requirements

The application shall adhere to the following quality attributes:

2.2.1 Performance

- **NFR5.1 Response Time:** The application shall respond to user interactions (e.g., checking in, loading meeting lists) within 2 seconds under normal network conditions.
- **NFR5.2 Bluetooth Transfer Speed:** Document transfers via Bluetooth should achieve a reasonable speed (e.g., average 100 KB/s for typical document sizes), considering the constraints of the technology.
- **NFR5.3 Battery Consumption:** The application's use of GPS and accelerometer should be optimized to minimize battery drain.

2.2.2 Security

- **NFR6.1 Data Encryption:** All sensitive user data, including location information and document content, shall be encrypted both in transit and at rest.
- **NFR6.2 Authentication:** Robust authentication mechanisms (e.g., password hashing, secure token management) shall be implemented.
- **NFR6.3 Authorization:** Users shall only have access to data and functionalities relevant to their roles (e.g., meeting organizer vs. participant).
- **NFR6.4 Privacy:** User location and activity data shall be handled in strict accordance with privacy regulations and user consent.

2.2.3 Reliability

- **NFR7.1 Availability:** The core functionalities of the application (meeting scheduling, attendance marking) shall be available 99.9% of the time, excluding scheduled maintenance.
- **NFR7.2 Data Integrity:** All attendance records, meeting details, and shared documents shall be stored accurately and consistently.
- **NFR7.3 Offline Capability (Partial):** The application shall support core functionalities like viewing previously downloaded meeting details and document sharing via Bluetooth even without internet connectivity.

2.2.4 Usability

- **NFR8.1 Intuitive User Interface:** The application shall have a clean, intuitive, and easy-to-navigate user interface.
- **NFR8.2 Error Handling:** The application shall provide clear and informative error messages to the user.
- **NFR8.3 Accessibility:** The application shall consider basic accessibility guidelines for users with disabilities (e.g., font sizes, color contrast).

2.2.5 Scalability

- **NFR9.1 User Load:** The system shall be able to support a growing number of users and concurrent meetings without significant degradation in performance.
- **NFR9.2 Data Storage:** The backend infrastructure shall be scalable to accommodate increasing volumes of attendance data and shared documents.

4 Architecture and Technical Options

4.1 Widget Tree Rationale

Our widget tree was structured to separate concerns cleanly and provide a scalable foundation for state-aware Flutter applications. We adopted a modular approach where each screen corresponds to a logical functional unit (e.g., login, registration, home, meeting, check-in).

The root widget initializes core services and wraps the app in a `MaterialApp` with route definitions. Navigation is handled using named routes for clarity and maintainability.

Each screen is broken into small composable widgets where necessary (e.g., list items, input forms). The use of BlocBuilder/BlocListener widgets allows UI to reactively update based on changes to the application's state.

The tree follows an MVVM-inspired separation: UI widgets present data, while BLoC components (as ViewModels) handle logic and Supabase interactions.

4.2 State Management

We adopted the **BLoC (Business Logic Component)** pattern using the `flutter_bloc` and `equatable` packages to ensure a predictable and scalable state architecture.

Each major functionality (e.g., user authentication) is managed by its own BLoC. For example:

- `UserBloc`: Manages user authentication state and session data.
- `MeetingBloc` (if implemented): Tracks meetings and check-in actions.

Each BLoC defines:

- Events (e.g., `LoginRequested`, `Logout`, `UserRegistered`)
- States (e.g., `UserAuthenticated`, `UserUnauthenticated`, `UserLoading`)

This ensures separation of logic from UI and facilitates testing and debugging.

4.3 Backend and Services

The entire backend infrastructure is built using **Supabase**, which provides:

- **Authentication:** Users register/login via email and password. Supabase handles token management and session persistence.
- **Database:** A PostgreSQL database stores all user, meeting, group, and attendance data.

The following key tables were defined in Supabase:

- `profiles` – stores user information (name, photo, etc.)
- `groups` – study groups created by users
- `group_members` – linking users to groups
- `meetings` – meeting metadata (title, date, location)
- `attendance` – linking users and meetings with a boolean `attended` flag
- `group_invites` – managing group join requests (if used)

All access is protected using Supabase Row-Level Security (RLS) policies, enforcing correct access based on the user session token.

4.4 Libraries and Packages Used

Below is a list of key packages used:

- `flutter_bloc` – state management with BLoC pattern
- `equatable` – value comparison for states and events
- `supabase_flutter` – integration with Supabase Auth and Database
- `flutter_secure_storage` – secure local storage for session persistence
- `geolocator` – GPS access and location validation
- `flutter_toast` – user feedback via toast messages
- `permission_handler` – runtime permission management

4.5 Other Technical Considerations

- **GPS Validation:** Attendance check-ins require proximity to the scheduled location. Geolocator is used to compare user coordinates with meeting location.
- **Offline Handling:** While most actions require connectivity, meeting data is cached locally and check-ins could be adapted for offline-first behavior.
- **Session Persistence:** User login state is maintained across app restarts using Supabase and secure storage.
- **Security:** All backend calls are authenticated with Supabase JWTs. RLS ensures users can only access their own data.
- **Scalability:** The BLoC architecture and database schema support future features like notifications, chat or analytics with minimal refactoring.

5 Overall Assessment

5.1 Achieved Objectives

The following project objectives were successfully implemented:

- Secure user authentication and registration using Supabase.
- Study group creation, joining by code, and viewing.
- Scheduling meetings with date, time, and GPS location.
- GPS-based attendance check-in and tracking.
- Attendance history display.
- Points-based gamification and leaderboard.
- Persistent user sessions across app restarts.
- Modular architecture using BLoC pattern.

Partially or Not Implemented Yet:

- Accelerometer-based presence detection (planned).
- Bluetooth document sharing (planned).
- Badges and notification system (planned).

5.2 Issues Found and Solutions

During development, we encountered several technical and organizational challenges:

- **Supabase Session Persistence:** Ensuring consistent session restoration across app restarts required use of `flutter_secure_storage` to persist JWT tokens.
- **GPS Permissions:** Managing runtime location permissions across devices and platforms involved handling edge cases where the user denies permissions or disables GPS.
- **BLoC Synchronization:** Synchronizing the UI with the state of the BLoCs required careful handling of transitions to avoid unnecessary rebuilds or inconsistent states.
- **Real-Time Updates:** Supabase Realtime was considered but deferred due to complexity. Meeting lists are instead refreshed manually or on navigation.
- **Team Coordination and Merge Conflicts:** Conflicts occasionally occurred when integrating features from different members. We resolved this through frequent communication and branching discipline in Git.

Lessons Learned:

- Effective use of BLoC significantly improves scalability and testability.
- Supabase offers a powerful full-stack solution but requires careful planning for permission rules and data modeling.
- Early definition of navigation flow and state architecture avoids rework.

5.3 Other Resources

GitHub Repository: `https://github.com/CarolineRibeiro19/studysync`

6 Contribution Assessment

6.1 Individual Contributions

- **Caroline Ribeiro:** [X%] – [Summary of contribution]
- **Mateus da Fonte:** [Y%] – [Summary of contribution]
- **Victor Milhomem:** [Z%] – [Summary of contribution]

7 Contribution Assessment

7.1 Individual Contributions

The project was collaboratively developed by three team members, and the overall workload was evenly distributed. Each member contributed approximately 33.3% to the final product. Below is a detailed enumeration of responsibilities, grouped by focus area and corresponding functionality.

7.2 Feature-Level Contributions

Caroline Ribeiro – Backend and Infrastructure

- Supabase configuration: tables `groups`, `group_members`, `group_invites`, `meetings`, `attendance`
- Creation of `GroupService` and `MeetingService` for database operations
- Secure queries and permission handling using Supabase RLS
- Logic for group invite codes and unique identifiers
- Integration foundation for GPS and accelerometer features

Mateus da Fonte – Frontend and UI/UX

- Implementation of key screens: `GroupScreen`, `GroupDetailScreen`, `LocationPickerScreen`
- Dialogs for group creation, joining by code, and meeting scheduling
- Map integration using Google Maps and visual display of locations
- Layout design for responsiveness and mobile usability
- Display logic for meetings, formatted dates, and group data

Victor Milhomem – State Management and Logic Integration

- Development of BLoC components: `MeetingBloc`, related events and states
- Connecting BLoC events to UI and services: `LoadMeetings`, `AddMeeting`, `LoadGroupMeetings`
- Synchronization of app state with interface updates via `BlocBuilder` and `BlocListener`
- Handling of loading indicators, error feedback, and empty data states

8 Tutorial / Manual

8.1 Getting Started

To run the application locally:

1. Clone the repository from GitHub:
   `git clone https://github.com/CarolineRibeiro19/studysync`
2. Install dependencies:
   `flutter pub get`
3. Configure Supabase credentials:
   Create a file named `supabase_config.dart` and insert your Supabase URL and anon/public key as constants.
4. Run the application on an emulator or physical device:
   `flutter run`

Requirements:

- Flutter SDK (>=3.0)
- Android Studio or VS Code with Flutter extension
- A Supabase project configured with the correct database tables

8.2 Main Features Walkthrough

Login and Registration Upon opening the app, users are directed to the login screen. New users can register via the Register button, which leads to a registration form requiring name, email and password. Successful registration redirects the user to the Home Screen.

Returning users simply enter their credentials to log in via Supabase Auth.

Home Screen After login, users land on the Home Screen. This screen shows:

- A list of today's meetings (if any)
- Bottom navigation bar with links to other sections: Home, Groups, Check-in
- A floating button to create or join groups

Group Management From the Groups section, users can:

- Create a new study group by giving it a name and subject
- Join an existing group using an invitation code
- View a list of all groups they belong to

Tapping on a group navigates to the Group Detail Screen, where upcoming and past meetings are shown.

Meeting Creation and Viewing Within the Group Detail Screen, organizers can:

- Tap "Schedule Meeting" to open the meeting creation screen
- Enter the meeting name, date/time and allow the app to auto-fill GPS location
- Submit the meeting, which is stored in Supabase and linked to the group

All members of the group will see the scheduled meetings listed.

Check-in Process From the Check-in tab:

- Users see a list of meetings scheduled for the current day.
- Tapping a meeting prompts the app to:
  - Validate if the user is near the scheduled GPS location
  - (Optional) Monitor accelerometer activity (feature in progress)
  - Mark presence by updating the `attendance` table

Successful check-ins are reflected in the user's attendance history and used for gamification scoring.

Gamification Each valid check-in adds points to the user's score. The system tracks:

- Points for presence
- Bonus for punctuality
- Streaks for consecutive days with attendance

These scores will appear in a ranking (leaderboard), motivating users through friendly competition.

9 Technical Specifications

This section details the technical environment in which the application was developed and is expected to run. It includes SDK versions, platform requirements, and minimum system configurations for compatibility.

9.1 Flutter and Dart Environment

- **Flutter SDK:** `>=3.1.0 <4.0.0`
- **Dart SDK:** Bundled with Flutter distribution
- **State Management:** BLoC Pattern (`flutter_bloc`, `equatable`)
- **Main Packages:** Supabase, Geolocator, Secure Storage, Toasts, Permission Handler

9.2 Android Platform Configuration

- **Compile SDK Version:** 34
- **Target SDK Version:** 34
- **Minimum SDK Version:** 24 (Android 5.0 Lollipop)

9.3 Other Tools

- **IDE:** Android Studio or Visual Studio Code (Flutter plugin required)
- **Version Control:** Git (with GitHub integration)
- **Backend as a Service:** Supabase (Authentication + Database)

10 References

- Flutter Official Documentation: `https://docs.flutter.dev/`
- Supabase (Backend as a Service): `https://supabase.com/`
- flutter_bloc Package: `https://pub.dev/packages/flutter_bloc`
- equatable Package: `https://pub.dev/packages/equatable`
- supabase_flutter Package: `https://pub.dev/packages/supabase_flutter`
- geolocator Package: `https://pub.dev/packages/geolocator`
- flutter_secure_storage Package: `https://pub.dev/packages/flutter_secure_storage`
- permission_handler Package: `https://pub.dev/packages/permission_handler`
- flutter_toast Package: `https://pub.dev/packages/fluttertoast`